LBL-10523 C.2
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

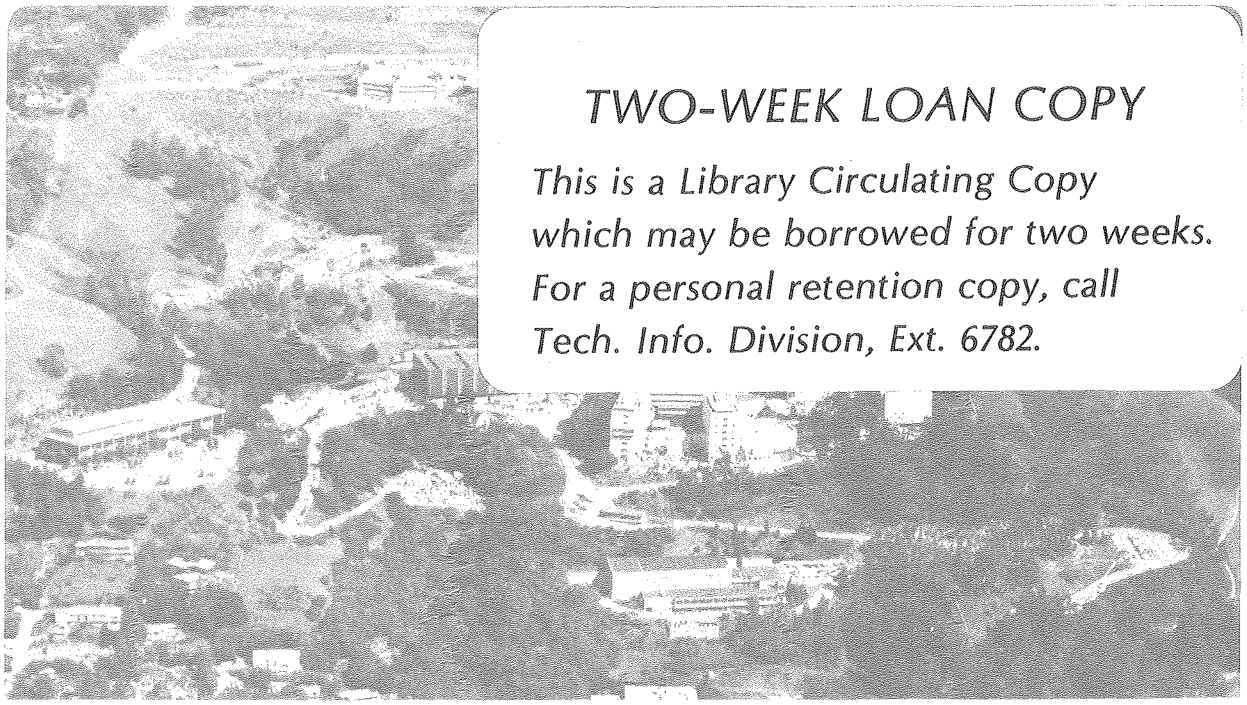
Engineering & Technical Services Division

Submitted to Computerworld

OBFUSCATORY MEASUREMENT: THE STATE OF THE ART

D. F. Stevens

February 1980



Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

LBL-10523

Submitted to Computerworld

Obfuscatory Measurement: The State of the Art*

D. F. Stevens

Lawrence Berkeley Laboratory
University of California
Berkeley, California

February, 1980

* This work was supported by the U. S. Department of Energy under contract No. W-7405-ENG-48.

Obfuscatory Measurement: The State of the Art*

D. F. Stevens
Lawrence Berkeley Laboratory
University of California
Berkeley, California

February, 1980

In its monumental issue introducing the 80's Computerworld (inadvertently, one hopes) failed to include a discussion of one of the principal trends in modern computer performance management, namely the flowering of obfuscatory measurement. As a whilom practitioner of that art, and as its principal (if not, indeed, its only) historian, I am pleased to have this opportunity to rectify that oversight. In this article I will first say a few words about the infancy of performance management and obfuscatory measurement, then look briefly at a few of the most popular current obfuscatory measures, discuss the foundations of obfuscatory measurement, and, finally, suggest some opportunities for the obfuscators of the future in the areas of distributed processing, data base applications, and word processing systems.

In the beginning

In the beginning was the Vendor, and the Vendor's word was unquestioned, and the Vendor's word was "move iron!" Performance management consisted of noticing that the work wasn't getting done and calling on the vendor for assistance. It is not really very surprising that that assistance usually took the form, after a suitable interval of scholarly-appearing activity,

* This work was supported by the U.S. Department of Energy under contract No. W-7405-ENG-48.

of additional hardware. It is somewhat surprising that the customers were so long in realizing that vendors exist to vend, and that perhaps an objective look at what was going on would be desirable.

But the questioning of authority is a contagious disease. Shortly after the DP departments begin questioning the pronouncements of the vendors, the users began questioning the pronouncements of the DP departments. The aroused user is a dangerous beast, capable of nearly superhuman feats.... Capable, even, of taking computing into her own hands should the provocation be sufficient. It was in the search for suitably soothing salves that obfuscation began, and the purpose of obfuscatory measurement has remained steadfast from that day to this: to divert the users' attention from the true state of the system with an imposing array of numbers, presented with a certain fervor and a modicum of quasi-religious awe.

The obfuscator is assisted in this task by the universal bureaucratic preference for numbers over judgement. ("Quality of service" is a subjective entity; one who presumes to judge it is subject to being overruled. "CPU utilization", on the other hand, is not a judgement but a determinable numeric quantity. [Better still, the means by which it is determined are under the control of the DP department.] It is, therefore, a Comfortable Construct, and hence much used in the world of obfuscation.) He--the obfuscator--can even call on Lord Kelvin for moral support: "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind."[1] A consequence of modern Kelvinism is the sense of heightened importance imparted to a quantity by the mere fact that it is being measured. Unmeasured qualities pale, unlamented, into insignificance when faced with detailed plots of interrupts per initiator or passwords per protocol.

One other contributory, perhaps accidental, to the development of obfuscatory measurement is the complexity of the systems measured. In the ideal world, one would be able to measure directly the quantities of presumed interest. In the world of computing those quantities are often well-protected and inaccessible. One must, instead, measure either their causes or their effects.

As indicated in Figure 1, the (inaccessible) interesting quantities are often the products of several (accessible) causes and give rise to (some aspects of) several (accessible) effects, but the relationships are quite diffuse. For example, one quantity which is generally thought to be interesting is the amount of user work passing through the system. That it is at present unmeasurable is unquestioned. So we measure various causal quantities (jobs, tasks, or sessions; overhead activities; queries; transactions; degree of multiprogramming) and various visible products (CPU and channel utilization; ABENDS; response time) and perform some unmentionable calculations therewith to produce our obfuscatory reports. And, indeed, it

[1]But the truly competent obfuscator is also aware of opposing views. This, from Daniel Yankelovich, for instance: "The first step is to measure whatever can be easily measured. This is okay so far as it goes. The second step is to disregard that which can't be measured or give it an arbitrary quantitative value. This is artificial and misleading. The third step is to presume that what can't be measured easily isn't very important. This is blindness. The fourth step is to say what can't be measured doesn't exist. This is suicide."

works: bamboozled by "97% success ratio" ((submissions - ABENDS)/submissions) and "83% saturated" (CPU utilization) the users go away knowing something is wrong but having no loose end to grasp.

<u>The most popular obfuscatory measures of the 70's (in alphabetical order)</u>

1. Availability

   Usually expressed as a percentage, "availability" is taken by the uninitiated to indicate the amount of time the system is usable, whereas in fact it indicates the amount of <u>scheduled</u> time the system is available <u>to the computer center</u>. By reducing the base to scheduled time a significant increase in percentage is obtained. It is further increased by including many periods of time when the system is not, in fact, fully usable: start-up times, time spent re-running lost or interrupted jobs, and time devoted to the "run-down" before a scheduled interruption. Figure 2 illustrates the cumulative effect of all these adjustments. It shows a week in the life of a one-shift operation, with one period of preventive maintenance (PM), a daily system development shot (SD), two unscheduled periods of down-time (15 minutes on Tuesday and an hour on Friday); start-up requires half an hour, and "run-down" starts a half-hour before system development time and an hour before the end of the shift. Naive and obfuscatory measures stand in rather sharp contrast.

2. Average Response Time

   This has superseded "turnaround time" as the most commonly quoted measure of turnaround, but the principles of use are the same. Its obfuscatory nature depends essentially upon the fact that the mean can easily be manipulated by stacking the extremes. To be specific, you can achieve essentially any average response time you wish by requiring a suitable number of trivial interchanges--with zero response time--to take place during any interactive session. The obfuscator also has a great deal of flexibility in the definition of the event that "response time" is monitoring. I have seen it variously defined as

- acknowledgement of the command/request
- commencement of the process
- first character of (process) output
- last character of (process) output

The first and third of these are most in keeping with the obfuscatory art; the third is especially so if the process is designed to give an instantaneous preliminary response.

Another fact to be borne in mind is that, in some situations, response which is <u>too</u> quick creates tension, which causes errors...and errors lead to wasted work, thus bringing saturation (and hence the opportunity for growth) ever closer. (A better strategy, however, is to strive for consistently unexpected response time, whether it be quicker or slower than anticipated...but this is somewhat off the subject of this article.)

3. Channel Activity

A utilization measure, and thus inherently obfuscatory, channel activity also exploits certain limitations in hardware design. For while it appears to measure data traffic, it in fact merely measures the time a certain hardware flag (the "channel busy" flag) is set. The actual relationship between channel activity and data traffic can be quite complex, depending not only upon the speed of the attached devices but also upon the housekeeping tasks which utilize the channel. The data traffic is always less than indicated by the obvious calculation (device speed multiplied by channel active time): indeed, it can sink to well below 10% of that number.

4. Depth of Multiprogramming; Overlap

These two measures are grouped together not because they are thought to be equivalent (they are not), but because they address the same problem: a vague understanding on the part of upper management that some multiplicity[2] of processing is desirable. They make a good combination, not only because they obfuscate in different ways, but also because the two together give no more accurate a picture than either one singly.

Overlap is in fact somewhat less obfuscatory than depth of multiprogramming, for it measures the percentage of time that some amount of simultaneity is experienced; it does not, however, consider the level of simultaneity. (Thus two simultaneous processes are every bit as good as seven.) It may be this very touch of honesty, paradoxically, which makes overlap so useful as an obfuscatory measure.

Depth of multiprogramming, on the other hand, is pure obfuscation: it counts initiators instead of processes. In many shops, large values of depth of multiprogramming survive as tribute to the memory

[2]To an obfuscator, multiplicity is merely advanced duplicity.

salesperson's art, while all the jobs lie quiescent awaiting the pleasure of the Resource Manager or some other such system magus.

5. Efficiency

This is actually a vestige of the more distant past, but one which has validity in some contexts, and adds a certain panache to many performance measurement reports. It is often used in place of "utilization" (the two are identical in meaning). (I would advise against using them both to refer to the same quantity: such a juxtaposition might inspire tiresome questions. "CPU utilization" and "channel efficiency", on the other hand, provide a nice appearance of breadth.)

6. Lines of Code

This measure, being directed at human productivity, might be considered by some to be somewhat outside the scope of this article, but programmer performance _is_ an element of computer center performance and lines-of-code _is_ superbly obfuscatory. The reason for this is that it does, in fact, measure productivity of a kind.... The kind that will saturate your systems in a hurry.

A timid person might hesitate to use lines-of-code on the grounds that it is patently absurd (is the Beer Bottle Song ["One hundred bottles of beer on the wall...."] better than a Shakesphere sonnet? a limerick than a haiku? this article than the Gettysburg Address?), inasmuch as it ignores quality. Such a person severely underestimates the power of numbers to convince and confuse.

7. MTBI (Mean Time Between Interruptions

MTBF (the mean time between failures) is so well accepted as a reliability measure in engineering contexts that practically no one questions its DP analog, MTBI. That the causes of failure in the two fields are largely unrelated is largely ignored: failures in mechanical systems are caused by wear and fatigue (to which software is impervious); failures in computing systems are caused by unexpected input (to which mechanical systems are rarely exposed) and trivial overflows (which, if they cause damage at all in mechanical systems, cause trivial damage: will an overflow on the meter crash a taxi?). The user-oriented measure which most closely corresponds to MTBI is the mean (or median) service interval. To see how they compare, we return to the sample week of Figure 2. The mean service interval, even giving full credit for the run-down periods, is 2.23 hours (26.75/12), and the median is 2.5. The conservative way to calculate MTBI is to divide "hours availble" by "number of interruptions plus 1": 32.75/3 = 10.9 hours...more than three times as long as the longest service interval.

8. Saturation

The obfuscatory nature of "saturation" lies in the fact that saturation is not a measure but a binary condition: the change in the quality of a service which moves from an unsaturated condition to a saturated one is an abrupt discontinuity: service effectively stops and the input queue becomes infinite. (We have all seen that happen with expressway rush-hour traffic.) References to "80 percent of saturation" thus really mean "80 percent of capacity", and are doubly obfuscatory because "capacity" changes with workload and environment. It is not a configuration constant; any reasonable multiprogramming system, for instance, has a smaller capacity when restricted to highly compute-bound jobs than when fed a mixture of compute- and I/O-bound work. The obfuscator exploits this phenomenon in other ways; it is much less well-known, for example, that any multiprogramming system strongly dominated by priority considerations has a smaller capacity than a system free to assign requested resources (such as the CPU) in an optimal fashion. (Is it any wonder that priority-dominated scheduling is so popular?)

9. Turnaround Time

Since the good turnaround times are the small ones, this is a situation where the median, surprisingly enough, favors the obfuscator. Nevertheless, I recommend sticking with the mean. For not only is the median a dangerous precedent to set, the mean is, as we have seen above, quite a tractable index. As in the case of response time the enterprising manager can cause enough small jobs to be submitted to achieve whatever mean turnaround time is deemed necessary. If this fails to provide the desired result, in desperation one can always define turnaround time in CPU terms, thus avoiding the semi-infinite delays of many print queues.

10. Utilization

When the obfuscator is asked for measures of throughput she has ample industry precedent for responding with measures of utilization. Utilization measures are advantageous because they reward ineffective programming (which is much easier to obtain than the other kind). The obfuscatory path here is not quite as free as it used to be, what with the introduction of distinguishable "system" and "problem" states for

CPU utilization...but it remains the case (thanks to your friendly mainframe vendor) that much of what is called "problem state" is actually system overhead. And it seems extremely unlikely that anyone is going to come up with a meter which distinguishes between "system" and "problem" channel activity states!

The Fundamentals of Obfuscation

Having seen the list of the ten best obfuscatory measures of the seventies, you should be able to pick out the most likely newcomers for the eighties. My selections follow, but first a quick resume of the underlying principles of obfuscatory measurement.

1. Select your measures with care.

   Not all measures are appropriate to all situations. You should neither attack the fly with the cannon, nor the elephant with the feather-duster. Tailor your measures to the tractability of your users and the gullibility of your upper management...and always have a couple in reserve, just in case.... In particular, your measures should be expressed in units which are well understood by your staff, and over the consumption (or generation, as appropriate) of which they have rather complete control. In so doing you create a climate in which improvements in the measurements are practically assured. You can also utilize the more traditional motivators: 100% CPU utilization can be guaranteed, for example, by telling your system supervisor that her pay will be her basic salary multiplied by the average CPU utilization for the month.

2. Seek the advice of your mainframe vendor.

Remember, the vendors were the first great obfuscators, and they remain members in good standing in this august fraternity. Furthermore, your vendor holds your interests close to his heart, for he cannot sell you additional equipment until your upper management is convinced of the saturation and effective utilization of your existing configuration. Furthermore, your vendor has a wealth of experience in dealing with upper managements just like yours.... Obfuscation is the very essence of the salesperson's art; as you seek legal advice from a lawyer, you should seek obfuscatory advice from your vendor.

3. Use the easiest measures.

The easier a measure is to obtain the more likely it is to be obfuscatory. (This is a rare _favorable_ instance of Murphy's Law.) Two particular kinds of easy measures are worth special consideration: _means_ and _percentages_. As we saw in the discussion of Availability, suitable definition of the base can turn any measurement into a praiseworthy percentage. As for the mean, it frequently lacks meaning. Even though the recent literature has exposed the obfuscatory nature of "indiscriminate" use of statistical concepts, the mean is so beloved by the average person that its utility is expected to continue relatively undiminished. One can still, for instance, report a favorable mean in preference to a realistic median in most circumstances. It is generally useful, in fact, to ignore all such distributional details as peaks, extremes, modal values, and repetitive and seasonal patterns in favor of the universal mean.

4. Exploit comfortable analogies.

Concepts which are meaningful in other fields can sometimes be transferred into the computer performance arena, where they are invalid, without loss of prestige. It helps, of course, if the concept is so familiar that it is accepted without question in its new context. MTBI is such a measure.

5. Pick evocative names for your measures.

   The creative definition of measurement jargon is an indispensable element of the obfuscator's arsenal...for the most misleading percentage you can devise won't help you unless you can convince someone that it measures <u>something</u>. If yours is an elementary situation, actual definition is not important: a catchy name is all that is required.... (Remember "CPU efficiency"? was there anything efficient about it? A modern example is "depth of multiprogramming".)

   If you find yourself in deeper waters, some measure of definition must be supplied...but it is best if it is either ambiguous or incompletely specified. ("Availability" as "percentage of time available" is, as we saw, an excellent example of this technique.)

<u>Obfuscatory Measures of the Future</u>

The most fruitful areas for the development of new obfuscatory measures are those portions of the DP universe which have caught the public fancy but for which there is no common agreement on terminology. In today's world, Data Base, Distributed Processing, and Word and Text Processing would seem to be the prime candidates, with Executive Information Systems coming along rapidly. On the grounds that something should be left as an exercise for the reader, I will not undertake to predict likely obfuscatory EIS measures, but will content myself with a few guesses in the other areas.

1. Word and Text Processing

a. Number of words in the spelling corrector

This measure is somewhat outside the mainstream of this article, being a bit of vendor obfuscation unlikely to see much use in the dialogue between the system guru and the user. But is perhaps worth recording if only to show that the vendors continue to break new obfuscatory ground. It is, of course, intended to prevent deeper inquiry into the spelling system: How much does spelling correction cost in time and space? How many of those words will I never use? How much of my company's idiosyncratic vocabulary is missing? How hard is it to add new words? Delete existing words? How many of them are misspelled?

b. Documents per day

This is the text-processing equivalent of job and session counts. The obfuscator need only provide a suitably fluid definition of document to ensure that her productivity figure of merit will show a gratifyingly steady upward trend. In this connection it is useful to note that many word- and text-processing systems have facilities for inter-office mail. One should perhaps exercise patience here, and add messages to the document count only when the pressure for productivity reaches fever pitch.

c. Keystrokes per hour

If you plan to use this one, you'd better be prepared to replace space and backspace keys rather frequently, and you might as well order systems with no tab facilities. (Why pay for a feature the operators will ignore? Of course you can counter that ploy by giving credit for tabbed-over blanks...at the cost of mid-page indentation....)

d. Response time

This old favorite will measure the time from last keystroke to the appearance of the first page of the first copy of the document, thus sidestepping the true issue of concern (document delay time).

e. (To be supplied later)

This is another exercise for the reader, for I cannot figure out how the practicing obfuscator can dodge the issue of the appearance of the document. My guess is that he will stonewall it or try to smother it under tons of productivity data. After all, time is on his side.... Within another generation no one will remember the pride once taken in individually formatting documents on the basis of amount of text, purpose, and intended audience. (Sigh.)

2. Distributed Processing

a. Message volume

As usual, activity will be quantified instead of qualified. Concern over the content and information density will be submerged in a welter of statistics on messages per hour, per node, or per node-hour. Intricate diagrams of internodal traffic volume will be used to overwhelm those concerned about possible mislocation of files and data. Volume can then be increased by simply reducing the maximum allowable information content per message. This can often be done in the name of reliability.

b. Reliability measures

As long as possible, reliability measures will be strictly hardware-component-oriented; that is, reliability statistics will be available on a node and link basis, rather than on a complete transaction basis. This makes it possible to quote quite impressive average reliability achievements even at a time when the users are seeing essentially zero reliability. For example, if you are dealing in 100-character messages over a two-hop path, a character reliability of 99.5% can be achieved while end-to-end message reliability drops to barely more than 3%. (There are five steps in a two-hop path: two links and three nodes. 99.5% on the character level can translate into a one-character error in half the messages at each step...i.e. the probability of successfully negotiating each step can be as low as .5; the probability of completing the journey can thus be as low as [(.5)**5] = .03125.)

While on this topic, it is well to note that the obfuscator has a choice of character-level or message-level reliability measures. He should choose the one which best complements his error-pattern. The key to the decision is the clumping tendency of the errors. If errors tend to occur in bursts, then the character-error-to-message-error ratio is high, and one should report message errors. If errors tend to be isolated, on the other hand, the character-error-to-message-error ratio is low, and you may wish to consider reporting on character reliability instead. (The exact conditions under which this becomes desirable are best left to the obfuscator's discretion, inasmuch as a proper choice depends upon the sophistication and docility of the users as well as upon the error patterns.)

c. Step counts; complexity

A favorite ploy, early in the game, will be the "Can you top this?" game, played with link and node counts. The number of steps negotiated will become more important than the manner in which they are negotiated. A perverse pride will even become evident in discussions of the number of transformations or translations to which the data must be subjected. Counts will reach incredible highs. As an example of what can be done even when the system is not very widely distributed, given suitable system architecture, consider the path of an execution module in a large-scale Control Data batch installation. Two computers are traversed, a front-end and a mainframe; each computer is comprised of peripheral processors (PPs) and a central processor. The path from card-reader to execution contains the following steps: card-reader, front-end PP, front-end buffer, PP, front-end queue, PP, front-end buffer, PP, mainframe PP, mainframe buffer, PP, mainframe queue, PP, mainframe for execution...some 14 nodes in all. When you describe such convoluted paths in full detail your users become grateful that _any_ messages get through unscathed.

d. Availability

The creative definition of availability will become both simpler and more necessary as systems become ever more distributed. Simpler, because one can adopt various component-oriented strategies. (A ten-element system, one element of which is always down, can be assigned nearly any availability score from 0 to 100% by simply assigning suitable weights to the components. The wise obfuscator will strive for weightings which give results in the 95-98% range.) Obfuscation will become more necessary because true availability may be severely impacted by the necessity to stop everything periodically to reconcile conflicting updates. I have no doubt the obfuscators of the Eighties will rise to the occasion.

3. Data Base

(Data base is, of course, a true natural for obfuscation, for its practitioners cannot even agree on the spelling of the name: data base, data-base, database.)

a. Data dictionary size

I believe that both extremes will see currency in this area. Some obfuscators will take pride in the manner in which they have reduced the number of distinct data elements to a minimum. (Perhaps some day one will receive the Turing Award for getting the number down to two.) Others will take equal pride in the flexibility of systems which allow hundreds, or even thousands, of distinct elements. The object, in either case, is to bewilder the users who want only--but all--of the data elements they use.

b. Number of queries

This is good because it allows even erroneous accesses to contribute to the productivity score. (Under no circumstances do we wish to count only those queries which were successful in the sense of giving the user the desired information in the desired form.)

c. Average response time, connect hours, number of enquiry stations, ...

In short, any obfuscatory measure which can be applied to any interactive system will be applied to data base systems. Because the user community will in many cases differ almost completely from the traditional DP user community the experienced obfuscator will experience little danger in making this transition.

d. Any measurement on a batch system

Any measurement on a batch data base system must be considered obfuscatory in the sense that it diverts the attention of the users away from the fact that they don't have an interactive system.

It seems unquestionable that the obfuscators among us will find the fields of the future to be as fruitful as the orchards of the past, and that they will continue to enliven our lives with (in the words of Pooh Bah) "corroborative detail, intended to add artistic verisimilitude to an otherwise bald and unconvincing narrative."

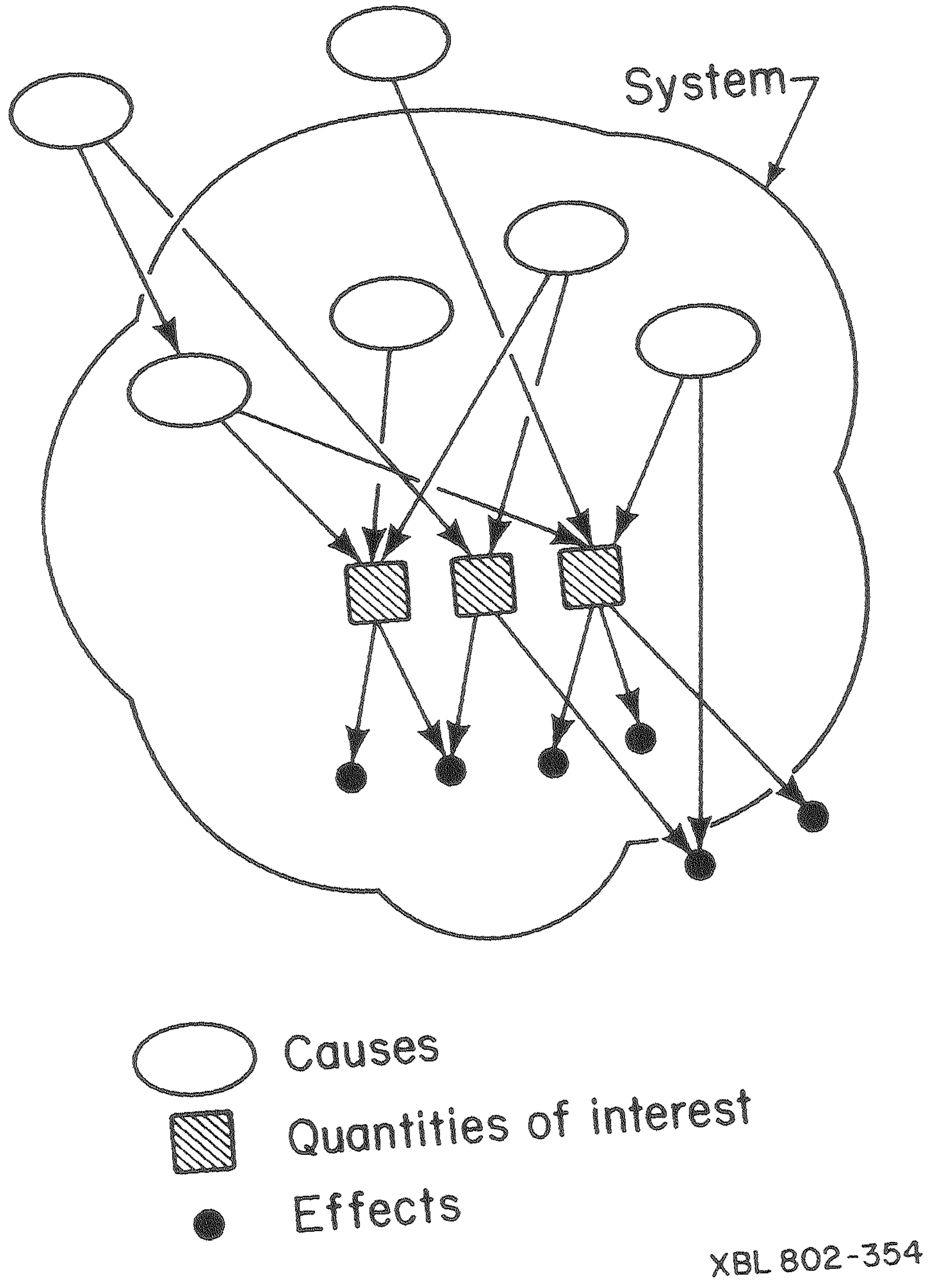
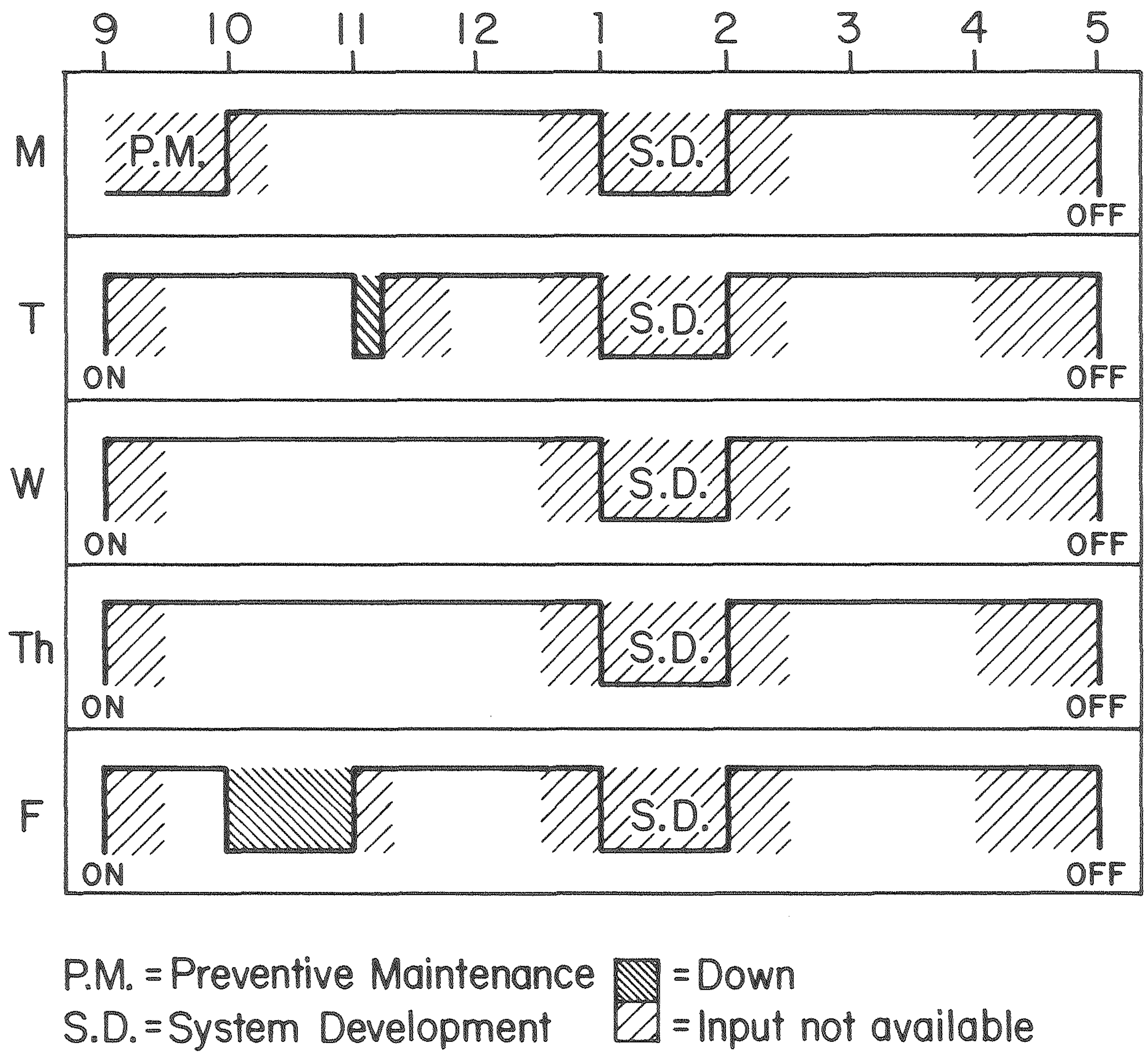


Figure 1: What are we measuring?



Obfuscatory (Avail/sched)	Naive (Avail/total)
6/6	3.5/8
6.75/7	3.75/8
7/7	4.5/8
7/7	4.5/8
6/7	3/8
32.75 / 34	19.25 / 40
96%	48%

XBL 802-355

Figure 2: Availability